Lista 2 - Econometria II

Guilhermo Argentieri

May 2025

Questão 2

(a)

A função objetivo do problema é definida como:

$$m(\beta, Y, X) := -\frac{1}{n}(Y - X\beta)'(Y - X\beta)$$

Obtendo as derivadas:

$$\nabla_\beta m(\beta, Y, X) = -\frac{1}{n}(2X'Y - 2X'X\beta) = -\frac{2}{n}X'(Y - X\beta)$$
$$\nabla^2_\beta m(\beta, Y, X) = -\frac{2}{n}X'X$$

Note que $\nabla_\beta m(\beta, Y, X)$ é uma função polinomial em β, enquanto $\nabla^2_\beta m(\beta, Y, X)$ é constante. Logo, ambas as derivadas existem e são contínuas, o que implica que $m(\beta, Y, X) \in \mathcal{C}^2$.

(b)

Do item anterior, temos que:

$$\nabla_\beta m(\beta, Y, X) = -\frac{2}{n}X'(Y - X\beta) \Rightarrow \nabla_\beta m(\beta_0, Y, X) = -\frac{2}{n}X'(\underbrace{Y - X\beta_0}_{:=e}) = -\frac{2}{n}X'e$$

Multiplicando por $\sqrt{n}$:

$$\sqrt{n}\nabla_\beta m(\beta_0, Y, X) = -\sqrt{n}\frac{2}{n}X'e = -\frac{2}{\sqrt{n}}X'e = -2\left(\frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_i'e_i\right)$$

Como $(X_i, Y_i) \sim_{i.i.d} (X, Y)$ e $e_i = Y_i - X_i\beta_0$ é uma transformação mensurável, os pares (X_i, e_i) também são $i.i.d$. Sabendo disso e das hipóteses do exercício 1, podemos mostrar que:

$$\mathbb{E}[X_i'e_i] = \mathbb{E}[\mathbb{E}[X_i'e_i|X_i]] = \mathbb{E}[X_i'\underbrace{\mathbb{E}[e_i|X_i]}_{=0}] = 0$$
$$\mathbb{V}[X_i'e_i] = \mathbb{E}[X_i'X_ie_i^2] - \mathbb{E}^2[\underbrace{X_i'e_i}_{0}] = \mathbb{E}[X_i'X_i\underbrace{\mathbb{E}[e_i^2|X_i]}_{\sigma^2}] = \sigma^2\mathbb{E}[X_i'X_i]$$

Por fim, podemos utilizar o TLC, garantindo que:

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n} X_i' e_i \xrightarrow{d} \mathcal{N}(0, \sigma^2 \mathbb{E}[X_i' X_i]) \Rightarrow \sqrt{n}\nabla_\beta m(\beta_0, Y, X) \xrightarrow{d} \mathcal{N}(0, \Sigma)$$

(c)

$$\Sigma = \mathbb{V}[-2X_i' e_i] = 4\mathbb{V}[X_i' e_i] = 4\sigma^2 \mathbb{E}[X_i' X_i]$$

(d)

A hipótese de homocedasticidade garante que a matriz de variância assintótica seja consistente e que os erros padrão dos estimadores sejam válidos. Sem homocedasticidade, a inferência estatística pode ser inválida.

Ademais, o posto completo de X assegura que $\mathbb{E}[X'X]$ é invertível, o que faz com que o estimador $\hat{\beta} = (X'X)^{-1}X'Y$ seja bem definido. Caso essa hipótese seja violada, a matriz não é invertível e o estimador não é identificável.

Questão 4

(a), (b) e (c)

```
#Definindo a função do enunciado
f_ex4 <- function(x) {
  (1/3)*x^4 - (1/2)*x^3 - (2/3)*x^2 - 1
}

#Função para aplicar o método de bracketing e encontrar raízes
busca_raizes <- function(inicio, fim, passo, tolerancia = 1e-4) {
  x_valores <- seq(inicio, fim, by = passo)
  fx_ex4_valores <- f_ex4(x_valores)

  intervalos_raiz <- list()
  raizes <- c()

  for (i in 1:(length(x_valores) - 1)) {
    if (fx_ex4_valores[i] * fx_ex4_valores[i + 1] <= 0) {
      intervalo <- c(x_valores[i], x_valores[i + 1])
      intervalos_raiz[[length(intervalos_raiz) + 1]] <- intervalo
      resultado <- tryCatch({
        uniroot(f_ex4, interval = intervalo, tol = tolerancia)$root
      }, error = function(e) NA)

      raizes <- c(raizes, resultado)
    }
  }

  return(list(
    passo = passo,
```

```
    intervalos = intervalos_raiz,
    tabela_fx = data.frame(x = x_valores, fx = fx_ex4_valores),
    raizes = sort(unique(na.omit(round(raizes, 5))))
  ))
}

#Defininco os passos para aplicar os itens (a), (b) e (c)
passos <- c(1, 0.5, 0.1)

for (p in passos) {
  resultado <- busca_raizes(-5, 5, p)

  cat("\n-----------------------------------------------\n")
  cat("Passo:", resultado$passo, "\n")
  cat("Intervalos com possível raiz:\n")
  print(resultado$intervalos)

  cat("Raízes encontradas (com tolerância = 1e-4):\n")
  print(resultado$raizes)
}
```

```
##
## -----------------------------------------------
## Passo: 1
## Intervalos com possível raiz:
## [[1]]
## [1] -2 -1
##
## [[2]]
## [1] 2 3
##
## Raízes encontradas (com tolerância = 1e-4):
## [1] -1.31990  2.49482
##
## -----------------------------------------------
## Passo: 0.5
## Intervalos com possível raiz:
## [[1]]
## [1] -1.5 -1.0
##
## [[2]]
## [1] 2.0 2.5
##
## Raízes encontradas (com tolerância = 1e-4):
## [1] -1.31988  2.49483
##
## -----------------------------------------------
## Passo: 0.1
## Intervalos com possível raiz:
## [[1]]
## [1] -1.4 -1.3
##
## [[2]]
```

```
## [1] 2.4 2.5
##
## Raízes encontradas (com tolerância = 1e-4):
## [1] -1.31991  2.49484
```

(d)

A principal diferença é que o método de bracketing utilizado nos itens a e b retorna o intervalo em que a raiz se localiza, enquanto o método utilizado no item c é capaz de identificar precisamente quais são as raízes.

Questão 5

(a)

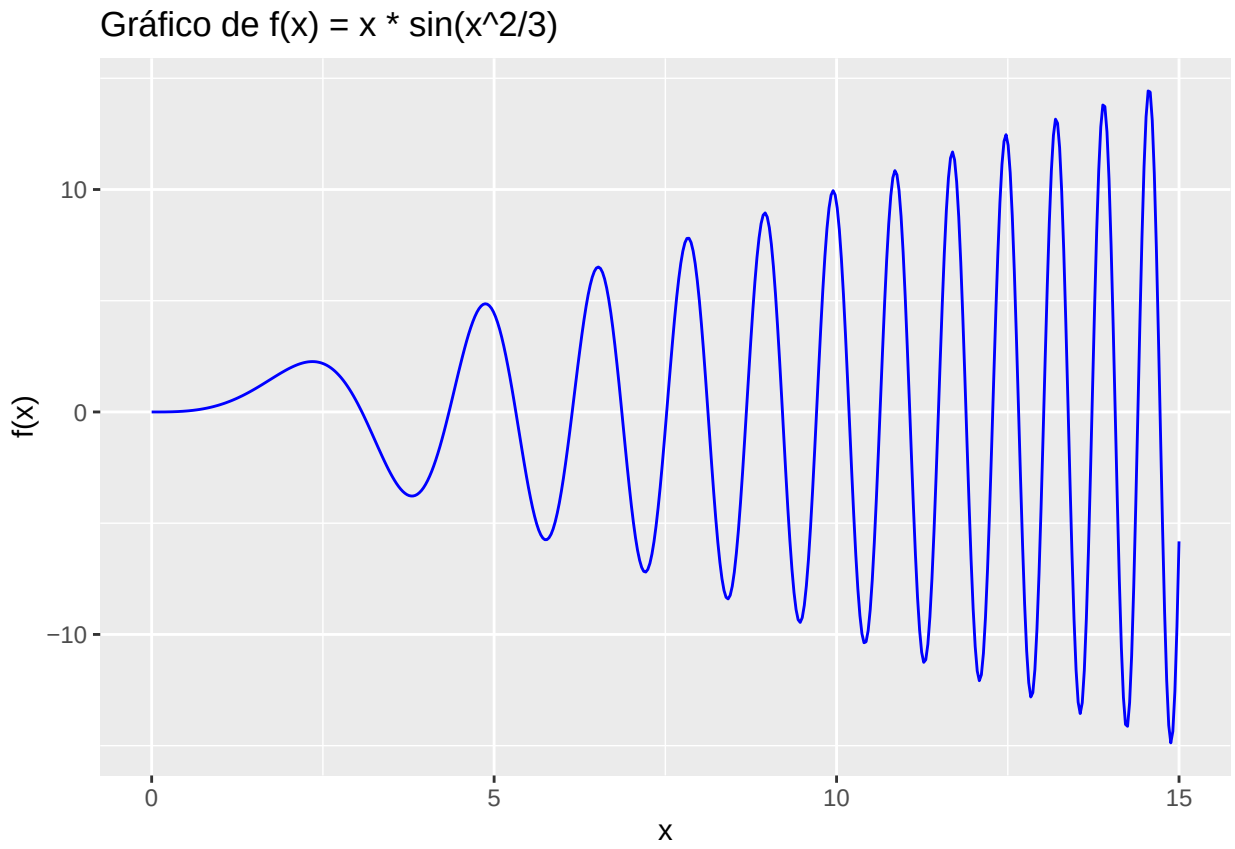
```
library(tidyverse)

#Definindo a função do enunciado
f_ex5 <- function(x) {
return(x*sin(x^2/3))
}

theta_0 = 0
lim_inf = 0
lim_sup = 15

#Método BFGS
resultado_bfgs <- optim(theta_0, fn = f_ex5, method = 'BFGS',
                        lower = 0, upper = 15, control = list(maxit = 100))

#Método SANN
resultado_sann <- optim(theta_0, fn = f_ex5, method = 'SANN',
                        lower = 0, upper = 15, control = list(maxit = 100))

#Método L-BFGS-B
resultado_lbfgsb <- optim(theta_0, fn = f_ex5, method = 'L-BFGS-B',
                          lower = 0, upper = 15, control = list(maxit = 100))

#Tabela de resultados
resultados_item_a <- tibble(metodo = c('BFGS','SANN', 'L-BFGS-B'),
                            argmin = c(resultado_bfgs$par,
                                       resultado_sann$par, resultado_lbfgsb$par),
                            minimo = c(resultado_bfgs$value,
                                       resultado_sann$value, resultado_lbfgsb$value))

print(resultados_item_a)
```

```
## # A tibble: 3 x 3
##   metodo   argmin minimo
##   <chr>     <dbl>  <dbl>
## 1 BFGS          0      0
```

```
## 2 SANN          0      0
## 3 L-BFGS-B      0      0
```

(b)

```r
#Dados para plotar a função
x_ex5 <- seq(0,15, length.out = 500)
y_ex5 <- f_ex5(x_ex5)
df_funcao <- tibble(x = x_ex5, y = y_ex5)

#Gráfico
df_funcao %>%
  ggplot(aes(x = x_ex5, y = y_ex5))+
  geom_line(color = 'blue')+
  labs(x = "x", y = "f(x)", title = "Gráfico de f(x) = x * sin(x^2/3)")
```



A resposta não faz sentido visto que, graficamente, podemos notar que o mínimo não é 0, logo, os três métodos erraram o mínimo.

(c)

```r
#Definindo os pontos iniciais de cada método
pontos_iniciais <- 0:15

pontos_iniciais_bfgs <- c()
pontos_iniciais_lbfgs <- c()
pontos_iniciais_sann <- c()

#Iteração rodando os pontos iniciais
for(theta in pontos_iniciais){
pontos_iniciais_bfgs <- c(pontos_iniciais_bfgs,
                          optim(theta, fn = f_ex5, method = 'BFGS', lower = 0, upper = 15,
control = list(maxit = 100))$par)

pontos_iniciais_lbfgs <- c(pontos_iniciais_lbfgs,
                           optim(theta, fn = f_ex5, method = 'L-BFGS-B', lower = 0, upper = 15,
control = list(maxit = 100))$par)

pontos_iniciais_sann <- c(pontos_iniciais_sann,
                          optim(theta, fn = f_ex5, method = 'SANN', lower = 0, upper = 15,
control = list(maxit = 100))$par)
}

resultado_item_c <- as_tibble_col(pontos_iniciais_bfgs, 'BFGS') %>%
cbind(as_tibble_col(pontos_iniciais_sann, 'SANN')) %>%
cbind(as_tibble_col(pontos_iniciais_lbfgs, 'L-BFGS-B')) %>%
round(digits = 4)
print(resultado_item_c)
```

```
##       BFGS    SANN L-BFGS-B
## 1   0.0000  0.0000   0.0000
## 2   0.0010  0.0010   0.0010
## 3   0.0011  0.0011   0.0011
## 4   3.8010  3.8010   3.8010
## 5   3.8010  3.8010   3.8010
## 6   8.4113  8.4113   8.4113
## 7   5.7552  5.7552   5.7552
## 8  12.0878 12.0878  12.0878
## 9  11.2814 11.2814  11.2814
## 10  9.4650  9.4650   9.4650
## 11 12.8437 12.8437  12.8437
## 12 12.8437 12.8437  12.8437
## 13 13.5576 13.5576  13.5576
## 14 12.8437 12.8437  12.8437
## 15 14.8830 14.8830  14.8830
## 16 14.8830 14.8830  14.8830
```

Questão 6

(a)

```
#Definindo a função do enunciado
f_ex6 <- function(x1, x2, theta) {
  theta1 <- theta[1]
  theta2 <- theta[2]
  theta3 <- theta[3]
  theta4 <- theta[4]
  theta5 <- theta[5]
  theta1*x1 + theta2 / (1 + exp(-theta3*x2)) + theta4*x1^2
}

#Valores do enunciado
data_points <- list(
list(x = c(1, 1), y = 35.8),
list(x = c(2, 4), y = 547.6),
list(x = c(-1, 2), y = 32.2),
list(x = c(2, -2), y = 14.5)
)

#Criando a função g
g <- function(theta) {
sum(sapply(data_points, function(pt) {
x1 <- pt$x[1]
x2 <- pt$x[2]
y_obs <- pt$y
(f_ex6(x1, x2, theta) - y_obs)^2
}))
}
```

(b)

```
#Verificando g(0,0,0,0)
theta_zero <- c(0, 0, 0, 0)
g_zero <- g(theta_zero)
cat("g(0,0,0,0) =", g_zero, "\n")
```

```
## g(0,0,0,0) = 302394.5
```

(c)

```
resultado_g <- optim(theta_zero, fn = g, method = 'BFGS', control = list(trace = 1))
```

```
## initial  value 302394.490000
## final  value 115121.522485
## converged
```

```r
theta_chapeu <- resultado_g$par
g_final <- resultado_g$value
iteracoes <- resultado_g$counts

cat("Melhor theta:", theta_chapeu, "\n")
```

```
## Melhor theta: 59.15143 131.6401 20.07176 17.06285
```

```r
cat("Valor mínimo de g:", g_final, "\n")
```

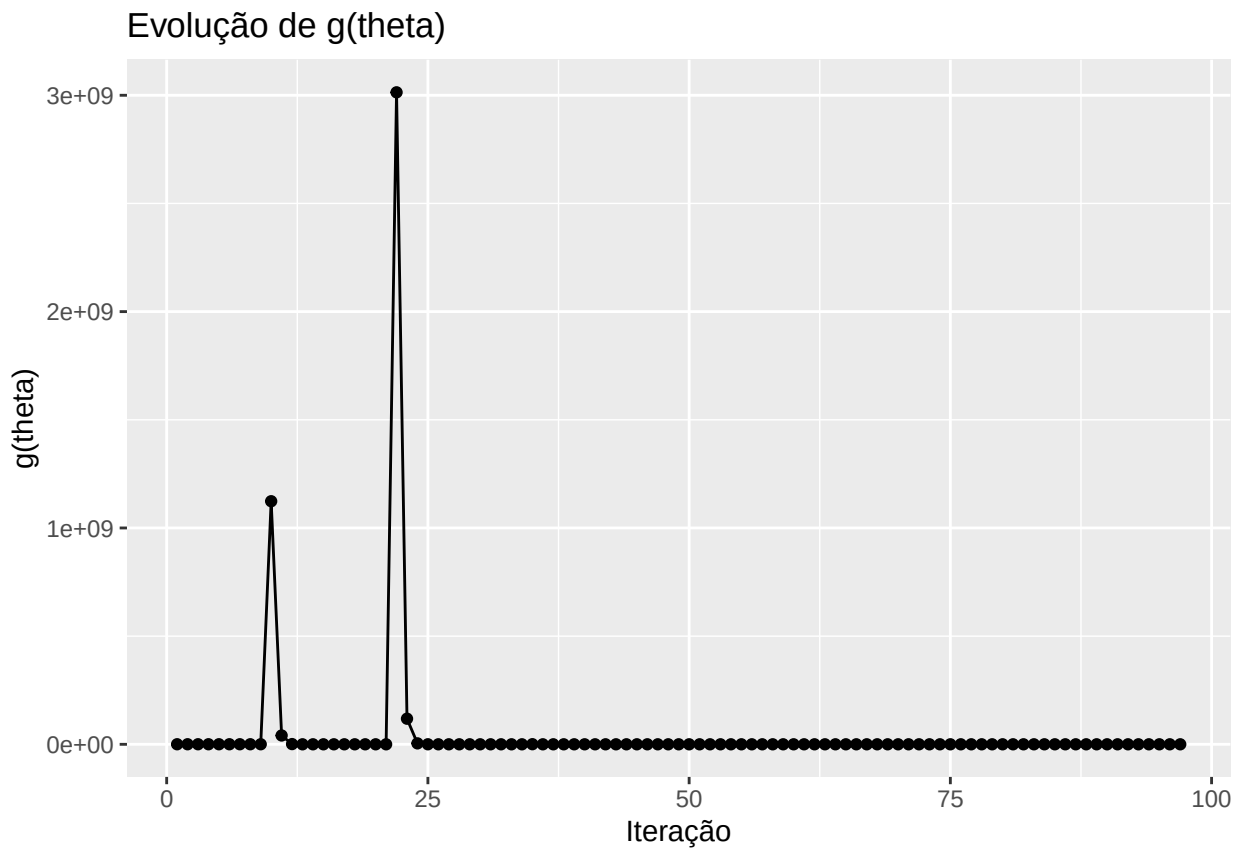
```
## Valor mínimo de g: 115121.5
```

```r
cat("Número de iterações:", iteracoes["function"], "iterções")
```

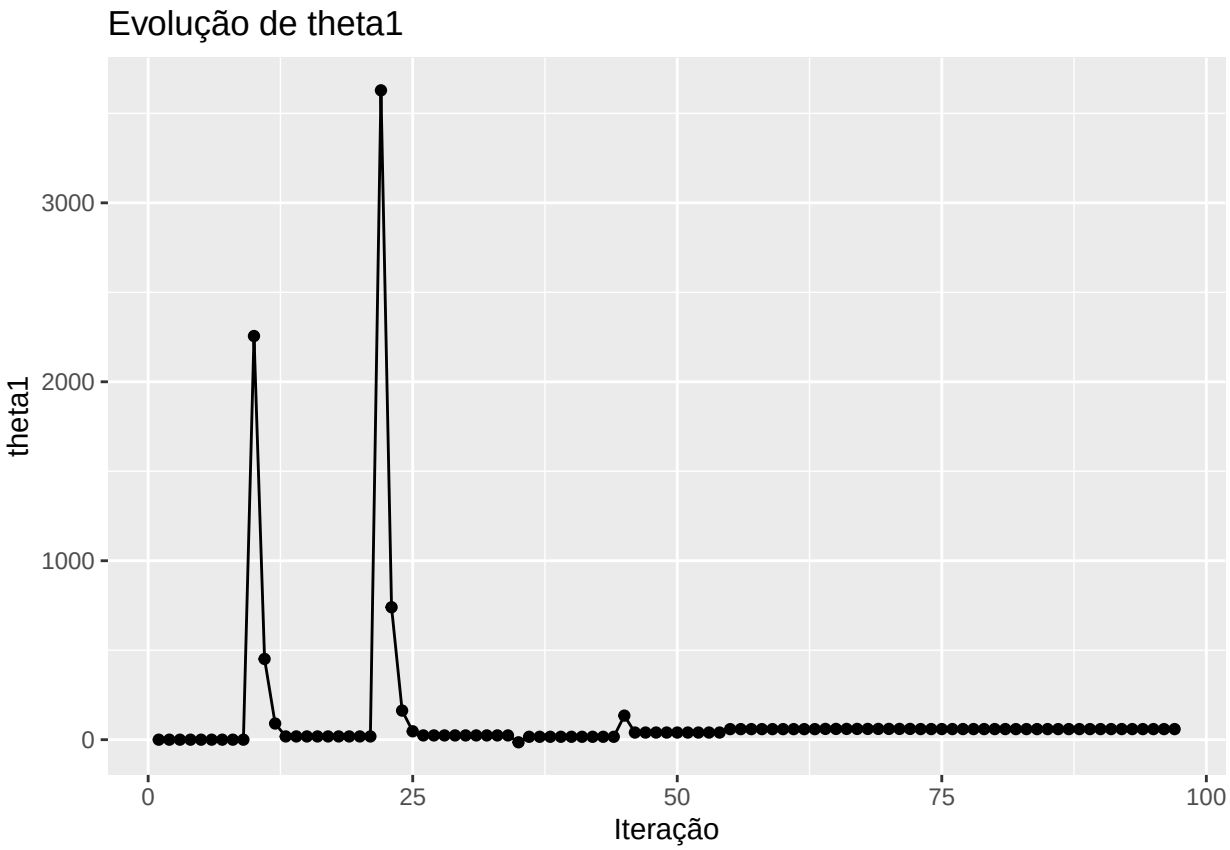
```
## Número de iterações: 25 iterções
```

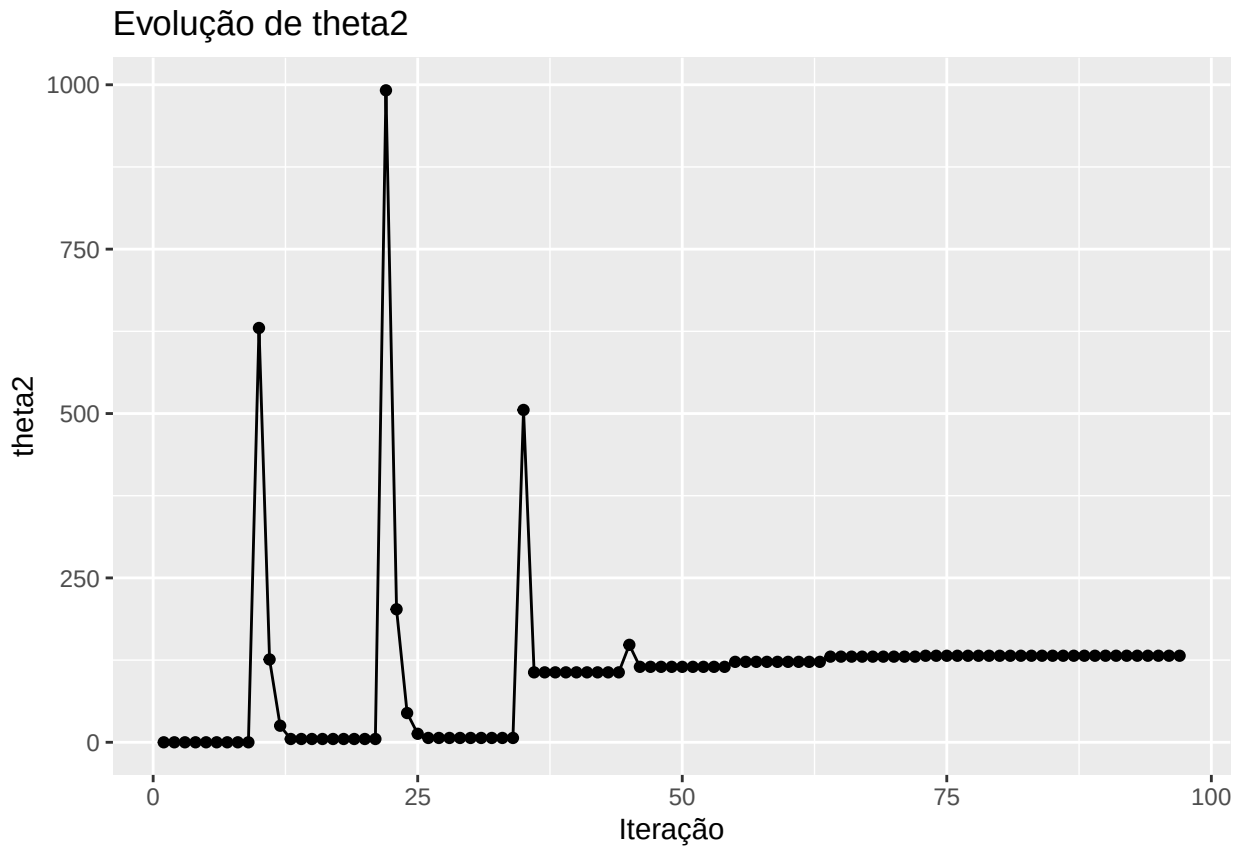
(d)

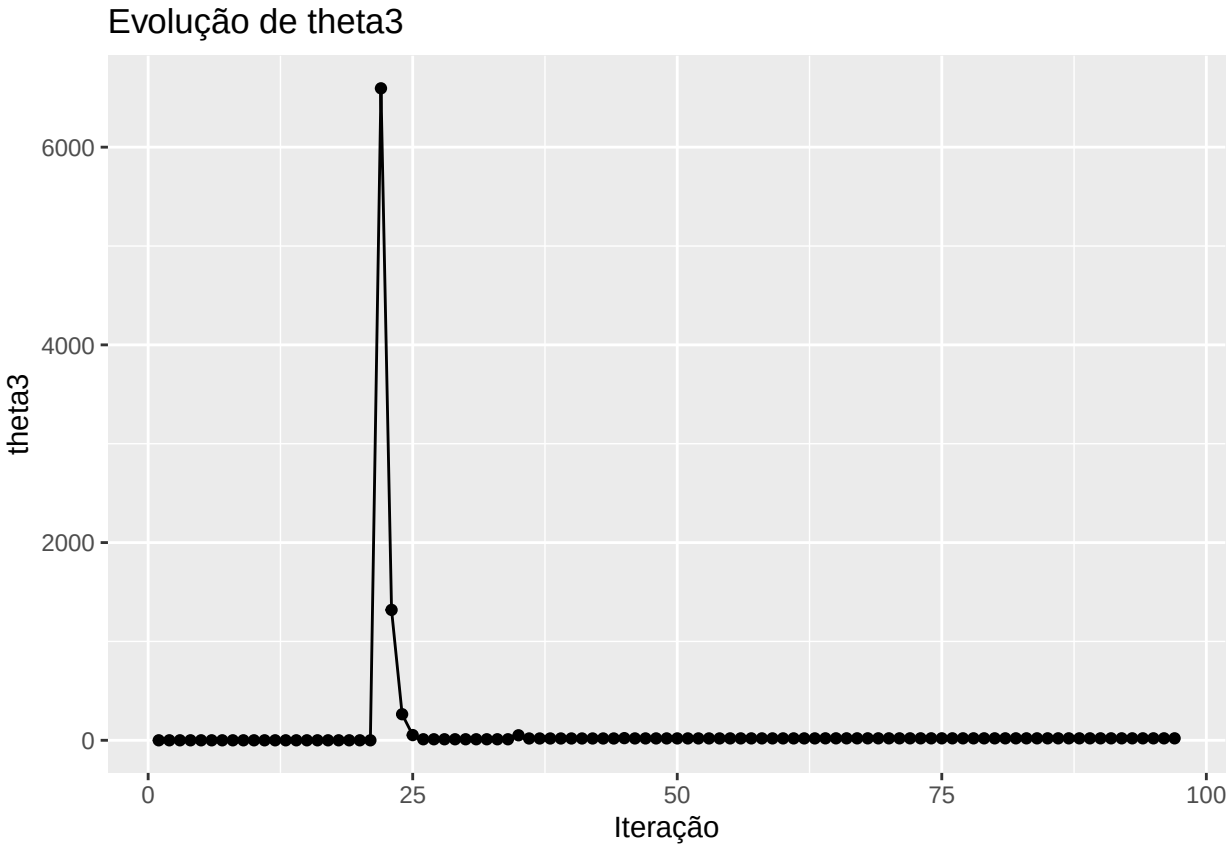
```r
#Lista para armazenar o histórico de valores de theta e g
historico_theta <- list()
historico_g <- numeric()

#Redefinindo g
g_com_rastro <- function(theta) {
  historico_theta[[length(historico_theta) + 1]] <<- theta
  valor_g <- g(theta)
  historico_g <<- c(historico_g, valor_g)
  return(valor_g)
}

#Reaplicando a otimização com a nova g
resultado_otimizacao <- optim(
  theta_zero,
  fn = g_com_rastro,
  method = "BFGS"
)


df_iteracoes <- as.data.frame(do.call(rbind, historico_theta))
colnames(df_iteracoes) <- c("theta1", "theta2", "theta3", "theta4")
df_iteracoes$g <- historico_g
df_iteracoes$iteracao <- seq_len(nrow(df_iteracoes))

#Gráfico 1: g(theta) por iteração
ggplot(df_iteracoes, aes(x = iteracao, y = g)) +
  geom_line() + geom_point() +
  labs(title = "Evolução de g(theta)", x = "Iteração", y = "g(theta)")
```

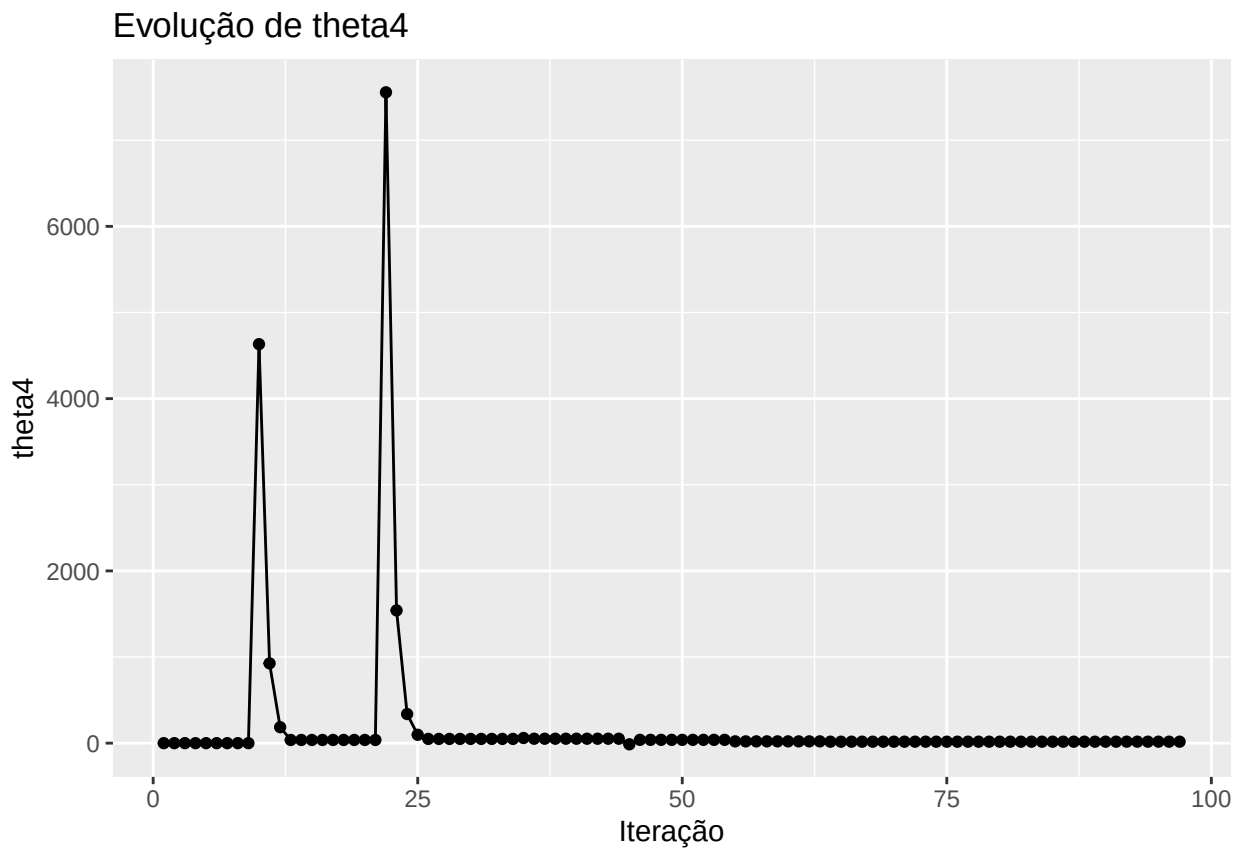



```r
#Gráficos para cada parâmetro
for (j in 1:4) {
  nome_coluna <- paste0("theta", j)
  grafico <- ggplot(df_iteracoes, aes_string(x = "iteracao", y = nome_coluna)) +
    geom_line() + geom_point() +
    labs(title = paste("Evolução de", nome_coluna), x = "Iteração", y = nome_coluna)
  print(grafico)
}
```


Evolução de theta1
theta1
3000
2000
1000
0
0
25
50
75
100
Iteração


Evolução de theta2
1000
750
500
250
0
theta2
0
25
50
75
100
Iteração


Evolução de theta3
6000
4000
2000
0
theta3
0
25
50
75
100
Iteração



Questão 9

(a)

```
#Definindo a semente
set.seed(123)

#Definindo os parâmetros e variáveis
beta_0 <- 3
beta_1 <- 4
n <- 100000
x <- runif(n, 0, 1)
epsilon <- rnorm(n, 0, 1)
y <- beta_0 + exp(beta_1*x) + epsilon

resultado_9a <- tibble(x = x, y = y)

head(resultado_9a)
```

```
## # A tibble: 6 x 2
##        x     y
##    <dbl> <dbl>
## 1 0.288   6.42
```

```
## 2 0.788  27.3
## 3 0.409   7.41
## 4 0.883  36.4
## 5 0.940  45.9
## 6 0.0456  6.46
```

(b)

```
nls_b <- nls(y ~ b_0 + exp(b_1*x), data = resultado_9a, start = list(b_0 = 1, b_1 = 1))

summary(nls_b)
```

```
##
## Formula: y ~ b_0 + exp(b_1 * x)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## b_0 3.0073421  0.0039711   757.3   <2e-16 ***
## b_1 3.9996327  0.0002332 17148.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.002 on 99998 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 1.716e-07
```

Há convergência em 6 iterações e está coerente com o processo gerador de dados, já que as estimativas estão muito próximas dos valores de β_0 e β_1 utilizados no item a.

(c)

```
#Criando a função
sqr <- function(par) {
b0 <- par[1]
b1 <- par[2]
y_chapeu <- b0 + exp(b1 * x)
sum((y - y_chapeu)^2)
}

sqr_min <- optim(par = c(0,0), sqr, list(method = 'BFGS'))

#Resultados dos betas
print(sqr_min)
```

```
## $par
## [1] 3.008618 3.999537
##
## $value
```

```
## [1] 100398.4
##
## $counts
## function gradient
##       95       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Há convergência, além disso, a estimativa está coerente, visto que assim como no item anterior, os resultados estão muito próximos aos valores de β_0 e β_1 utilizados no item a.

(d)

```r
nls_d <- nls(y ~ 3 + exp(b_1*x), data = resultado_9a, start = list(b_1 = 50))
summary(nls_d)
```

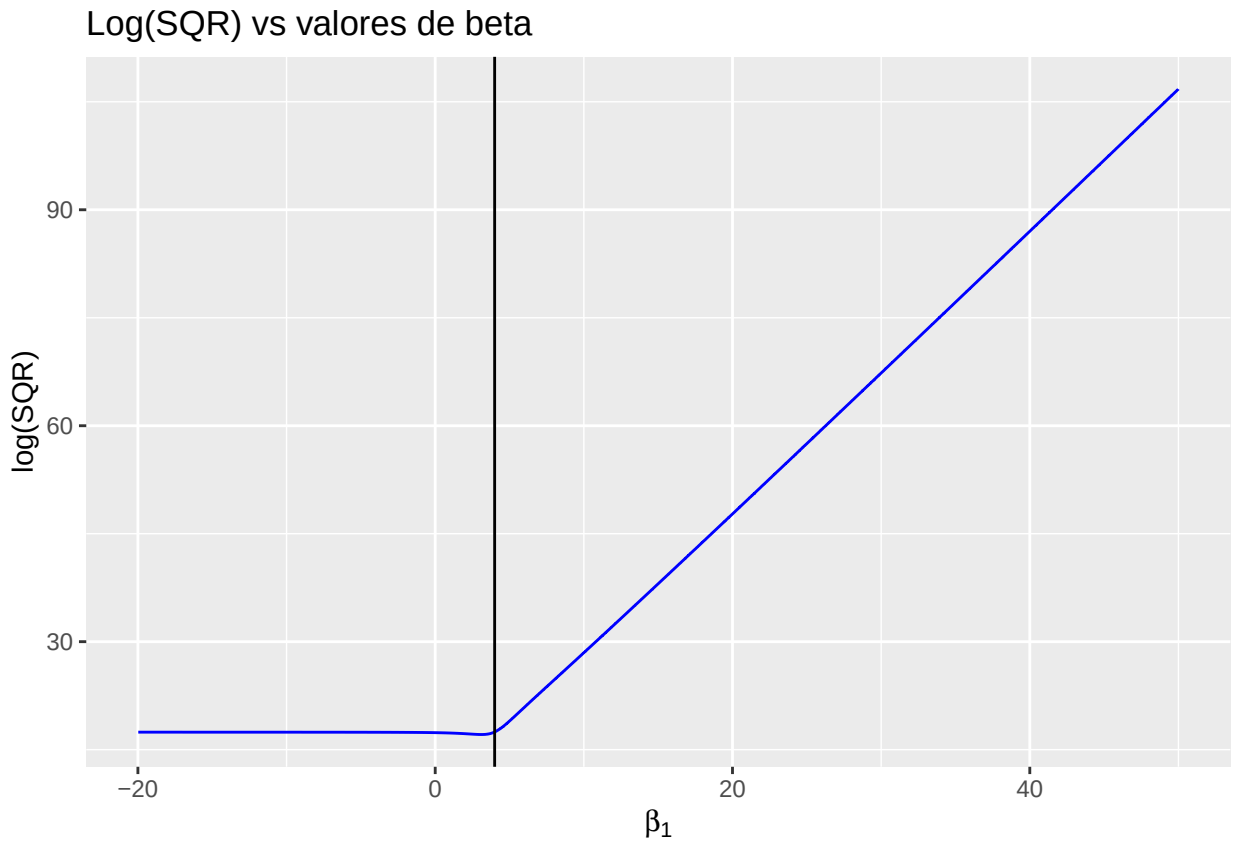
```
##
## Formula: y ~ 3 + exp(b_1 * x)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## b_1 3.9998926  0.0001861   21498   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.002 on 99999 degrees of freedom
##
## Number of iterations to convergence: 49
## Achieved convergence tolerance: 1.627e-09
```

Novamente, há convergência, ademais, houve convergência para o verdadeiro $beta_1$ visto que o resultado foi muito próximo de 4

(e)

```r
#Redefinindo as variáveis

x_e <- runif(10000, 0, 1)
epsilon_e <- rnorm(10000, 0, 1)
beta_0_e <- 3
beta_1_e <- 4
y_e <- beta_0_e + exp(beta_1_e*x) + epsilon_e

#SQR com b0 fixo
```

```r
sqr_beta_1 <- function(b1_val) {
y_chapeu <- beta_0_e + exp(b1_val * x_e)
sum((y_e - y_chapeu)^2)
}

beta_1_vals <- runif(10^4, -20, 50)
sqr_vals <- sapply(beta_1_vals, sqr_beta_1)

df_9e <- tibble(x = beta_1_vals, y = sqr_vals)

df_9e %>%
  ggplot(aes(x = x, y = log(y)))+
  geom_line(color = 'blue')+
  geom_vline(xintercept = 4)+
  labs(x = expression(beta[1]), y = "log(SQR)", title = "Log(SQR) vs valores de beta")
```



Parece que faz sentido, já que o ponto de mínimo do SQR está na vizinhança do valor verdadeiro de β_1.